Generování pseudonáhodných čísel po cifrách

Pavel Stříž (Malipivo)
pavel@striz.cz

OSSConf, Žilina, SK
2. – 4. července 2024

https://archive.org/details/2024-statisticke-dny-striz

Osnova

Pokusím se svými slovy okomentovat a přiblížit publikovaný článek

Generování pseudonáhodných čísel po cifrách

- ve sborníku OSSConf 2024, a,
- v mimořádném čísle roku 2024 Informačního bulletinu České statistické společnosti, `https://www.statspol.cz/informacni-bulletin/on-line-verze/`.

O problému

- V kombinatorice a programování (ranking-unranking problem) pracujeme s velkými čísly u výpočtů počtu možností, nesmíme tudíž zaokrouhlovat.
- Typickým příkladem je Rubikova kostka, byť jen pro $n = 3$ je počet možností N 20ciferné číslo.
- Pro $n = 2^{11}$ je počet cifer počtu řešení 16257517.
- Podobně se dostáváme do problémů u faktoriálů, byť v Pythonu můžeme užít `sys.set_int_max_str_digits(0)`. Problém je dělení, pokud si neoddělíme čitatele a jmenovatele, např.

```
$ python3
>>> import math
>>> math.factorial(52)
80658175170943878571660636856403766975
    289505440883277824000000000000
>>> math.factorial(52)/2*2
8.065817517094388e+67
>>> math.factorial(52)*0.5*2
8.065817517094388e+67
```

Už to chce knihovnu Decimal…

Jak obejít dělení: karetní hra bridž

$$\begin{aligned} N &= \binom{52}{13} \cdot \binom{39}{13} \cdot \binom{26}{13} \cdot \binom{13}{13} = \\ &= \frac{52!}{13!39!} \cdot \frac{39!}{13!26!} \cdot \frac{26!}{13!13!} \cdot \frac{13!}{13!0!} = \frac{52!}{13!^4} = \\ &= \frac{52 \cdot 51 \cdots 4 \cdot 3 \cdot 2}{13 \cdots 2 \ \cdot \ 13 \cdots 2 \ \cdot \ 13 \cdots 2 \ \cdot \ 13 \cdots 2} = \\ &= \frac{2 \cdot 2 \cdot 13 \ \cdot \ 3 \cdot 17 \ \cdots \ 2 \cdot 2 \ \cdot \ 3 \ \cdot \ 2}{13 \ \cdot \ 2 \cdot 2 \cdot 3 \ \cdots \ 2 \ \cdot \ \ldots} = \\ & \text{vše ve jmenovateli se vykrátí,} \\ & \text{pronásobíme zbylé činitele, 29 cifer,} \\ &= 53644737765488792839237440000. \end{aligned}$$

Generátory pseudonáhodných čísel

- Zde se dostaneme do výpočetních problémů ještě dřív. Náhodně generovat malá čísla není problém, ať už bity (0, 1) nebo cifry (0–9).
- Dostupným algoritmům podstrčit balíček typu Decimal není tak jednoduché.
- Proto se snažíme problém převést na práci s textovými řetězci.
- Soutěžní programování upouští od úloh s velkými čísly, zde má práce začíná. Reference: kniha *Competitive Programming 4*.

Tohle neprogramovat! Ošklivosti!

- Nehledět na interval, např. 0000–5563, generovat 0000–9999. U počtu možností ulítneme z intervalu, nepočítatelné u 5564–9999.
- Každou cifru brát zvlášť, např. u 235: 0–2 na 1. cifře, 0–3 na 2. a 0–5 na poslední. Vynecháme tím celou řadu čísel, např. 175 ad.
- Uříznout první cifru a pak generovat 0–9 na cifrách dalších. Např. z 000–356 se stane 00–99. Uřízneme 100–356. Můžeme uříznout až 90 % čísel.
- Odečíst 1 na první cifře, pak 0–9 na ostatních. Např. z 000–765 se stane 000–699. Uřízneme 700–765. 10 %+. Ne však víc jak 50 %.
- Následující cifry si hlídáme, ale např. u čísla 12, generovat 0 a 1 se stejnou pravděpodobností je chyba. Dostaneme se do problémů s podmíněnou pravděpodobností. Správně k 0 patří 0–9 (10 možností) a u 1 jen 0–2 (3 možnosti). Pod nulu na 1. cifře patří 10/13 situací, k jedničce jen 3/10 situací. U obřích čísel je to výpočetně extrémní, problém se zaokrouhlováním pravděpodobnosti. . .

Rychlé řešení

Vygenerovat číslo za použití 0–9, ale na konci to srovnat s horní mezí.

Např. u 0000–2278: 1245 je OK, 4899 není OK, 9000 není OK atd.

Tohle se běžně používá, ale je to pomalé, neb řadu čísel musíme vyřadit než se nám vygeneruje vhodné. Nemáme však problém s podmíněnou pravděpodobností.

Úvaha / Navržený algoritmus

- Pokud bychom generovali číslo v intervalu $[0; N-1]$ po cifrách, co nám v tom brání? Např. $N = 4383$, 0000–4382.
- Pokud je cifra menší než na 1. pozici, 0–3, další cifry na zbylých pozicích mohou být libovolné, tedy 0–9.
- Pokud je cifra stejná, 4, musíme kontrolovat další cifry, ale generování pokračuje.
- Pokud je cifra vyšší, 5–9, dostáváme se do podmíněné pravděpodobnosti, je tedy **nutné generování začít znovu.**
- Pozn. Někdy se místo $N-1$ klade M nebo $L = N-1$ (indexování od nuly).

Prototypování v Pythonu, po-cifrach.py

```
import random
hodnota=4383 # Indexování od 1 -- N.
cislo=str(hodnota-1) # Indexování od 0 -- N-1.
kolik=40 # Kolik hodnot si preji vygenerovat.
print("Generuji",kolik,"hodnoty od nuly po",cislo,"...")
delka=len(cislo)
def generuj():
    while True:
        odCisla=0; retezec=""; doCisla=int(cislo[0])
        mensi=False; poradi=0; znovu=False
        while poradi<delka:
            nahodne=random.randint(odCisla,doCisla)
            doCisla=9
            if nahodne<int(cislo[poradi]): mensi=True
            if nahodne<=int(cislo[poradi]) or mensi:
                retezec+=str(nahodne)
            else:
                znovu=True; break
            poradi+=1
        if not znovu:
            nalezene=int(retezec)
            if nalezene<hodnota:
                print(retezec); return()
for _ in range(kolik): generuj()
```

Konverze mezi jazyky

- Z Pythonu do R jsem užil: https://www.javainuse.com/py2r
- Z Pythonu do JavaScriptu jsem užil: https://extendsclass.com/python-to-javascript.html
- Zaujal mě projekt https://www.codeconvert.ai/, ale je to komerční produkt.

Ukázka pro R, po-cifrach.R

```
# Generování PRN po cifrách...
hodnota <- 4383 # Indexování od jednièky, 1 -- N.
cislo <- as.character(hodnota - 1) # Indexování od nuly, 0 -- N-1.
kolik <- 40 # Pocet chtených cifer.
print(paste("Generuji", kolik,"hodnot(y) od nuly po", cislo, "..."))
delka <- nchar(cislo)
generuj <- function(){
  while(TRUE){
    odCisla <- 0; retezec <- ""
    doCisla <- as.numeric(substr(cislo, 1, 1))
    mensi <- FALSE
    poradi <- 0; znovu <- FALSE
    while(poradi < delka){
      nahodne <- sample(odCisla:doCisla, 1)
      doCisla <- 9
      if(nahodne < as.numeric(substr(cislo,poradi+1,poradi+1))){mensi <- TRUE}
      if(nahodne <= as.numeric(substr(cislo,poradi+1,poradi+1)) || mensi){
        retezec <- paste(retezec, nahodne, sep = "")
      } else { znovu <- TRUE; break }
      poradi <- poradi + 1
    }
    if(!znovu){
      nalezene <- as.numeric(retezec)
      if(nalezene < hodnota){
        print(retezec); return()
}}}}
for(invalue in 1:kolik){generuj()}
```

Ukázka generovaných hodnot

```
$ Rscript po-cifrach.R

[1] "Generuji 40 hodnot(y) od nuly po 4382 ..."
[1] "3591"
[1] "2481"
[...]
[1] "2795"
[1] "1070"
```

Základ HTML pro JavaScript

```
<!DOCTYPE html>
<html>
    <head>
        <script defer src="./nahodna-cisla.js"></script>
    </head>
</html>
```

nahodna-cisla.js

```
var cislo, delka, hodnota, kolik;
hodnota = 4383; cislo = (hodnota - 1).toString();
kolik = 40; console.log("Generuji", kolik, "hodnoty od nuly po", cislo, "...");
delka = cislo.length;
function generuj() {
  var doCisla, mensi, nahodne, nalezene, odCisla, poradi, retezec, znovu;
  while (true) {
    odCisla = 0; retezec = ""; doCisla = Number.parseInt(cislo[0]);
    mensi = false; poradi = 0; znovu = false;
    while (poradi < delka) {
      nahodne = Math.floor(Math.random()*doCisla)+odCisla;
      doCisla = 9;
      if (nahodne < Number.parseInt(cislo[poradi])) {mensi = true;}
      if (nahodne <= Number.parseInt(cislo[poradi]) || mensi) {
        retezec += nahodne.toString();
      } else { znovu = true; break; }
      poradi += 1;
    }
    if (!znovu) {nalezene = Number.parseInt(retezec);
      if (nalezene < hodnota) { console.log(retezec); return [];}
    }
  }
}
for (var _ = 0, _pj_a = kolik; _ < _pj_a; _ += 1) {generuj();}
```

Postřehy

- Proces je cca dvakrát rychlejší než když vygenerujeme celé číslo a pak srovnáváme s horní mezí (přijmeme či vyřadíme).
- Jeden výpočetní stroj může generovat první cifry a o zbytek si říct jiným strojům, ty generují cifry v intervalu 0–9.
- Obvykle zlom nastává v prvních několika desítkách cifer, více v článku pomocí simulace.
- Pozn. Práce s obřími čísly je stále otevřený problém v C/C++.

Editor SciTE

- R standardně nenabízí mezi jazyky v nabídce, umazal jsem ve filtru `r` na konci souboru `/usr/share/scite/SciTEGlobal.properties`.
- Spuštění přes klávesu F5 jsem nastavil v souboru `~/.SciTEUser.properties`: `command.go.$(file.patterns.r)=Rscript $(FileNameExt)`.
- Navíc jsem si přidal `user.shortcuts=Ctrl++|2333|Ctrl+-|2334|`, abych mohl zvětšovat a zmenšovat písmo přes Ctrl+plus a mínus i mimo numerickou klávesnici.

Mé další užití výpočetního prostředí R

`https://www.statspol.cz/informacni-bulletin/on-line-verze`

- 4/2019: PF2020! aneb Když nestačí ani R, ani TEX.
- 1/2020 (s Ondřejem Vencálkem): Mapa míst konferencí ROBUST.
- 3/2020: Novinky ze světa R+koronaviru.
- M/2021: Několik postřehů k Asymptote.

PF 2020! ČStS

stringr
googledrive
tidyverse
Shiny
tidyr
readr
consensus
R2D3

2020
letní
zimní
2020 Bardějov

bars Coronavirus Tracker

Confirmed Cases - Worldwide

3 1158040

COPY EMBED CODE

Select a country to see it displayed below.

Cases in Australia

7,000 6,000 5,000 4,000 3,000 2,000 1,000 0

01-22 2020, 01-30 2020, 02-20 2020, 03-12 2020, 04-02 2020, 04-23 2020, 05-03 2020

info_circle Home
waveform_p John Hopkins
waveform Weixin
graph_circle DXY
quote_bubb News

Petal.Length
Sepal.Width
Sepal.Length